Caso Policía Nacional

El departamento de Inteligencia Militar que está asociado a la policía nacional está trabajando después de la pandemia con el doble de casos debido a un aumento en casos especiales de inteligencia. Por esta razón, se han visto atrasados en muchos casos ya que el personal con el que cuenta no es suficiente para analizar toda la data que ellos explotan. Debido a esto, les interesa automatizar el proceso de extracción de información importante y palabras claves de los miles documentos que reciben diarios por fotografías tomadas por espías.

Toda esta información la almacenan en una carpeta local secreta en donde cada imagen se nombra bajo la sintaxis "caso_sec_img_xx.png".

Lo que necesitan es un sistema que extraiga el texto de una imagen y luego analice su contenido para retener sus palabras claves y atributos principales.

● Recursos locales que emplearán.

-Utilizamos Computadora de escritorio.

-Utilizamos una laptop.

-También estuvimos utilizando programas en nuestros equipos locales como Visual studio code.

-Lenguaje de Programación: Python.

-Almacenamiento local para guardar imágenes y documentos.

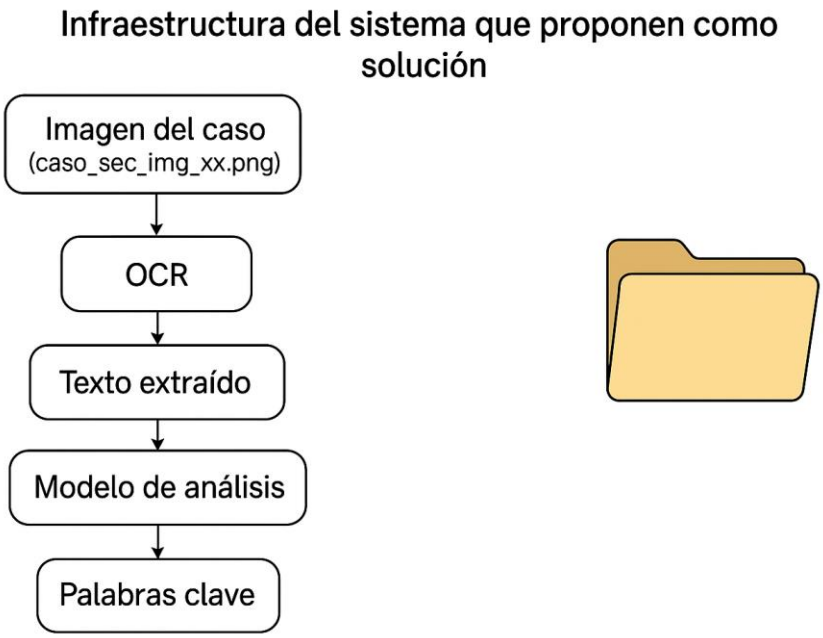
-fotografías en diferentes fuentes de Google.

Recursos en la nube que usarán. Módulos de Azure ML que emplearán y por qué.

-Utilizamos Computer Vision y TextAnality) para leer el texto desde la imagen.

-Se agregó tkinter → Para mostrar el texto extraído en una ventana emergente.

● Diagrama de flujo de los datos y proceso.



● Procedimiento realizado para resolver el problema.

-Usamos una herramienta OCR (como Azure Computer Vision y TextAnality) para leer el texto desde la imagen.

-Luego pasamos ese texto a un modelo de lenguaje para que encuentre las palabras importantes.

-Lo probamos con varias imágenes reales para asegurarnos de que funcione bien.

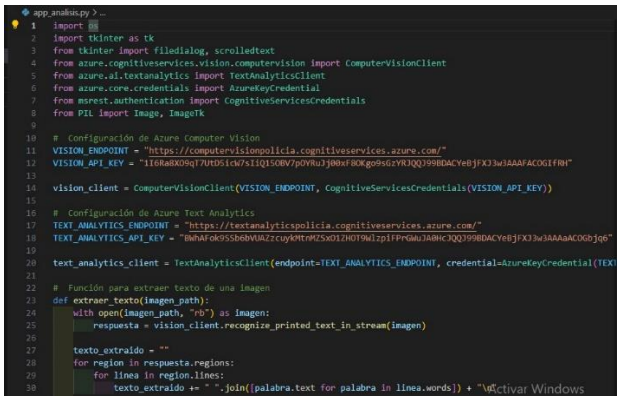
● Método de recolección de datos y recopilación de la información de entrenamiento.

-Utilizamos la misma imagen de nuestro proyecto, del caso policía nacional.

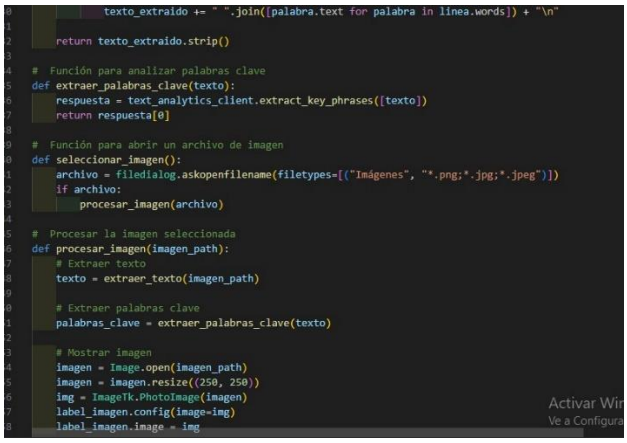
-También utilizamos una imagen de caso de policía nacional de diferente fuentes de Google.

● Formato de salida propuesto de la información.

Por consola y Terminal en texto.



```python
import os
import tkinter as tk
from tkinter import filedialog, scrolledtext
from azure.cognitiveservices.vision.computervision import ComputerVisionClient
from azure.ai.textanalytics import TextAnalyticsClient
from azure.core.credentials import AzureKeyCredential
from msrest.authentication import CognitiveServicesCredentials
from PIL import Image, ImageTk

#  Configuración de Azure Computer Vision
VISION_ENDPOINT = "https://computervisionpolicia.cognitiveservices.azure.com/"
VISION_API_KEY = "1I6RaBXO9q77UtD5icW7sIiQ15OBV7pOYRuJj00xF8OKgo9sGzYRJQQJ99BDACYeBjFXJ3w3AAAFACOGIfRH"

vision_client = ComputerVisionClient(VISION_ENDPOINT, CognitiveServicesCredentials(VISION_API_KEY))

#  Configuración de Azure Text Analytics
TEXT_ANALYTICS_ENDPOINT = "https://textanalyticspolicia.cognitiveservices.azure.com/"
TEXT_ANALYTICS_API_KEY = "BWhAFok9SSb6bVUAZzcuykMtnMZSxOI2HOT9Wlzp1FPrGWuJA0HcJQQJ99BDACYeBjFXJ3w3AAAaACOGbjq6"

text_analytics_client = TextAnalyticsClient(endpoint=TEXT_ANALYTICS_ENDPOINT, credential=AzureKeyCredential(TEXT

#  Función para extraer texto de una imagen
def extraer_texto(imagen_path):
    with open(imagen_path, "rb") as imagen:
        respuesta = vision_client.recognize_printed_text_in_stream(imagen)

    texto_extraido = ""
    for region in respuesta.regions:
        for linea in region.lines:
            texto_extraido += " ".join([palabra.text for palabra in linea.words]) + "\n"
```

```python
            texto_extraido += " ".join([palabra.text for palabra in linea.words]) + "\n"

    return texto_extraido.strip()

#  Función para analizar palabras clave
def extraer_palabras_clave(texto):
    respuesta = text_analytics_client.extract_key_phrases([texto])
    return respuesta[0]

#  Función para abrir un archivo de imagen
def seleccionar_imagen():
    archivo = filedialog.askopenfilename(filetypes=[("Imágenes", "*.png;*.jpg;*.jpeg")])
    if archivo:
        procesar_imagen(archivo)

#  Procesar la imagen seleccionada
def procesar_imagen(imagen_path):
    # Extraer texto
    texto = extraer_texto(imagen_path)

    # Extraer palabras clave
    palabras_clave = extraer_palabras_clave(texto)

    # Mostrar imagen
    imagen = Image.open(imagen_path)
    imagen = imagen.resize((250, 250))
    img = ImageTk.PhotoImage(imagen)
    label_imagen.config(image=img)
    label_imagen.image = img
```


```python
def procesar_imagen(imagen_path):
    imagen = imagen.resize((250, 250))
    img = ImageTk.PhotoImage(imagen)
    label_imagen.config(image=img)
    label_imagen.image = img

    # Mostrar resultados en la interfaz
    text_resultado.delete(1.0, tk.END)
    text_resultado.insert(tk.END, f" Texto Extraído:\n{texto}\n\n")
    text_resultado.insert(tk.END, f" Palabras Clave: {', '.join(palabras_clave)}")

#  Crear la interfaz gráfica
root = tk.Tk()
root.title("Análisis de Imágenes con IA ")
root.geometry("600x500")

frame = tk.Frame(root)
frame.pack(pady=10)

btn_seleccionar = tk.Button(frame, text=" Seleccionar Imagen", command=seleccionar_imagen, font=("Arial", 12))
btn_seleccionar.pack()

label_imagen = tk.Label(root)
label_imagen.pack(pady=10)

text_resultado = scrolledtext.ScrolledText(root, width=70, height=10, font=("Arial", 10))
text_resultado.pack(pady=10)

# Ejecutar la interfaz gráfica
root.mainloop()
```


● Rol de los miembros del equipo y trabajo realizado.

Yoelvy – Programación del análisis de texto:
A Yoelvy le toca la parte de escribir el código que se encarga de analizar el texto que se extrae de las imágenes. Básicamente, él va a hacer que el sistema entienda lo que dice el

texto, identificando las palabras más importantes, organizando la información y preparándola para que sea útil. Es como enseñarle al sistema a leer y a sacar lo más relevante de lo que lee.

Wandy – Configuración del OCR y pruebas con imágenes:
Wandy se está encargando de poner a punto el OCR, que es la herramienta que "lee" el texto en las imágenes. Él va a probar con diferentes tipos de imágenes (claras, borrosas, con diferentes fondos, etc.) para asegurarse de que el sistema pueda reconocer bien las letras sin importar cómo se vea la imagen. En resumen, él está afinando la vista del sistema para que no se le escape nada.

● Costos de la propuesta.

-Uso de servicios OCR en la nube: $30/mes (Azure o Google Cloud)

-Infraestructura computadora potente local: $500 único

-Desarrollo del software: hecho por el equipo costo cero.

● Métricas de efectividad del sistema propuesto.

Precisión del OCR	Porcentaje de caracteres correctamente identificados en comparación con el texto original.	≥ 95%
Tasa de Extracción de Texto Válido	Porcentaje de imágenes en las que se logró extraer texto útil.	≥ 90%
Exactitud en Identificación de Palabras Clave	Porcentaje de palabras clave verdaderamente relevantes que el sistema logra extraer.	≥ 85%
Tasa de Falsos Positivos	Palabras marcadas como clave pero que no lo son.	≤ 15%
Tasa de Falsos Negativos	Palabras clave que el sistema no detectó.	≤ 15%

Tasa de Éxito en Imágenes con Calidad Baja	Porcentaje de éxito en la extracción de texto en imágenes con baja calidad.	≥ 50%
Compatibilidad con Formatos Diversos	Capacidad para procesar imágenes en distintos formatos como .png, .jpg, etc.	Al menos 2 formatos comunes

● Guia de uso de la demo realizada.

-Vamos al apartado Analizar_Texto.

-Adjuntamos la imagen de nuestra carpeta secreta.

-Luego que nos arroje el texto de la imagen lo copiamos.

-Vamos al apartado de Analizar texto para que nos arroje las palabras claves.

● Dificultades al realizar el proyecto.

-Vivir en Latam, eso dificulta que haya luz y no podamos usar nuestros equipos.

-Reconocimiento de texto en imágenes de baja calidad

-Selección precisa de palabras clave

-Organización y pruebas con imágenes simuladas

● Posibles mejoras de la propuesta.

-Hacer que reconozca también texto escrito a mano.

- Manejo de errores → Si la API falla, muestra un mensaje claro.

-Verificación de credenciales → Si hay un problema con la clave o endpoint, avisa.

-Un entorno visual mas atractivo.

Riesgos y limitaciones.

-Si la imagen está muy borrosa, el OCR falla.

-Si el espía usa códigos o lenguaje oculto, la IA puede no entender.

-Hay riesgo de que alguien acceda a la carpeta secreta si no está bien protegida.